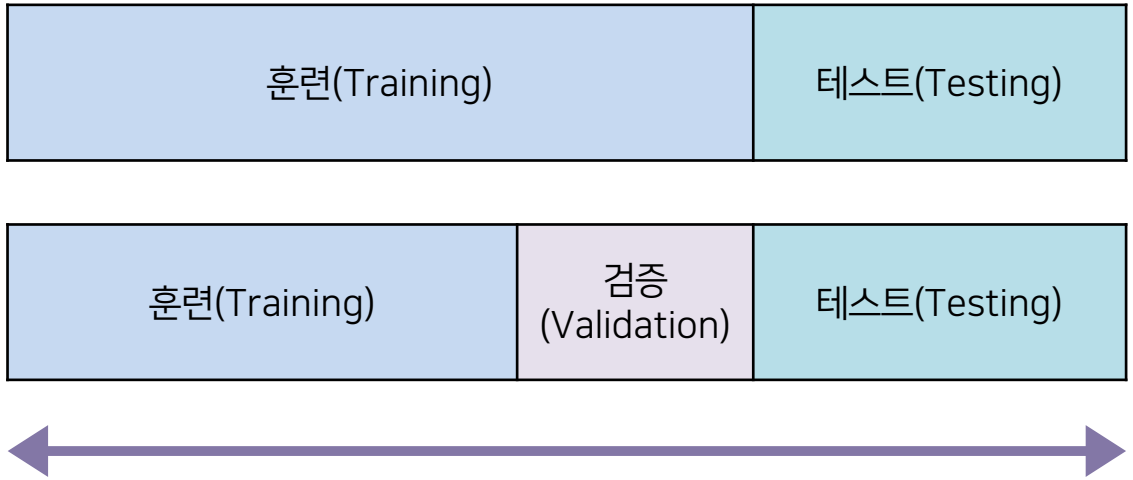
멋쟁이사자처럼 AI 스터디 1조

인공 신경망
(ANN)

김하영 박시윤 이성민

1 머신러닝 용어

(1) 머신러닝 모델의 평가



실제 모델의 평가를 위해서는 데이터를
훈련용, 검증용, 테스트용으로 분리하는 것이 일반적

- **검증용 데이터**: 모델의 성능 조정
  → 과적합 판단 및 하이퍼파라미터 조정
- **하이퍼파라미터(초매개변수)**: 모델의 성능에 영향을 주는 매개변수
- **매개변수**: 학습을 통해 바뀌어 가는 변수
  (예: 가중치, 편향)

초매개변수 vs 매개변수

- **초매개변수**
  - 사용자가 직접 정해줄 수 있는 변수
  - 경사하강법의 학습률(learning rate), 딥 러닝의 은닉층 수, 뉴런 수, 드롭아웃 비율 등
- **매개변수**
  - 모델이 학습하는 과정에서 얻어지는 값
  - 기계가 훈련을 통해 바꾸는 변수

1 머신러닝 용어

(2) 분류(Classification)와 회귀(Regression)

이진 분류 문제
(Binary Classification)

- 주어진 입력에 대해서 **둘 중 하나의 답**을 정하는 문제
- 시험 성적에 대해 합격 불합격을 판단하는 문제, 메일을 정상 메일인지 스팸 메일인지 판단하는 문제 등

다중 클래스 분류
(Multi-class Classification)

- 주어진 입력에 대해서 **세 개 이상의 선택지** 중에서 답을 정하는 문제
- 선택지를 카테고리 또는 범주, 클래스라고 함

회귀 문제
(Regression)

- 분류 문제처럼 분리된(비연속적인) 답이 아닌 **연속된 값을 결과로** 가짐
- 시계열 데이터를 이용한 주가 예측, 생산량 예측, 지수 예측 등

1 머신러닝 용어

(3) 지도 학습(Supervised Learning)과 비지도 학습(Unsupervised Learning)

지도 학습

- 레이블(Label)이라는 **정답(y, 실제값)**과 함께 학습
- 예측값과 실제값의 차이인 오차를 줄이는 방식으로 기계가 학습됨

비지도 학습

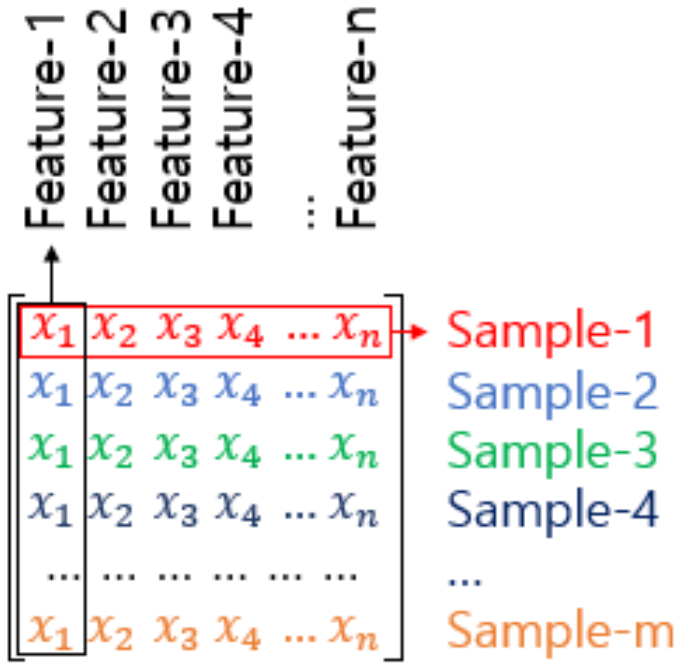
- **목적 데이터(또는 레이블)이 없는** 학습 방법
- 군집(clustering)이나 차원 축소와 같은 학습 방법

강화 학습

- 환경 내에서 정의된 에이전트가 현재의 상태를 인식하여 선택 가능한 행동 중 **보상을 최대화**하는 행동 선택

1 머신러닝 용어

(4) 샘플(Sample)과 특성(Feature)



- 훈련 데이터를 행렬로 표현
- 독립 변수의 개수가 n개이고 데이터의 개수가 m인 행렬 X

샘플(Sample)
하나의 데이터, 하나의 행

특성(Feature)
종속 변수 y를 예측하기 위한 각각의 독립 변수 x

1 머신러닝 용어

(5) 혼동 행렬(Confusion Matrix)

양성(Positive)과 음성(Negative)을 구분하는 이진 분류

		예측	
		참	거짓
실제	참	TP (True Positive)	FN (False Negative)
	거짓	FP (False Positive)	TN (True Negative)

- True: 정답을 맞힌 경우
- False: 정답을 맞히지 못한 경우
- Positive/Negative: 각각 제시했던 정답

- True Positive: 실제 Positive인 값을 Positive라고 예측(정답)
- True Negative: 실제 Negative인 값을 Negative라고 예측(정답)
- False Positive: 실제 Negative인 값을 Positive라고 예측(오답)
- False Negative: 실제 Positive인 값을 Negative라고 예측(오답)

정확도(Accuracy)

$$Accuracy = \frac{TP + TN}{TP + FN + FP + TN}$$

- 전체 데이터 중에서 제대로 분류된 데이터의 비율
- 정확도가 높을수록 잘 맞힌 모델

1 머신러닝 용어

(5) 혼동 행렬(Confusion Matrix)

양성(Positive)과 음성(Negative)을 구분하는 이진 분류

		예측	
		참	거짓
실제	참	TP (True Positive)	FN (False Negative)
	거짓	FP (False Positive)	TN (True Negative)

- True: 정답을 맞힌 경우
- False: 정답을 맞히지 못한 경우
- Positive/Negative: 각각 제시했던 정답

정밀도(Precision)

$$Precision = \frac{TP}{TP + FP}$$

- 모델이 Positive라고 분류한 데이터 중에서 실제로 Positive한 데이터의 비율

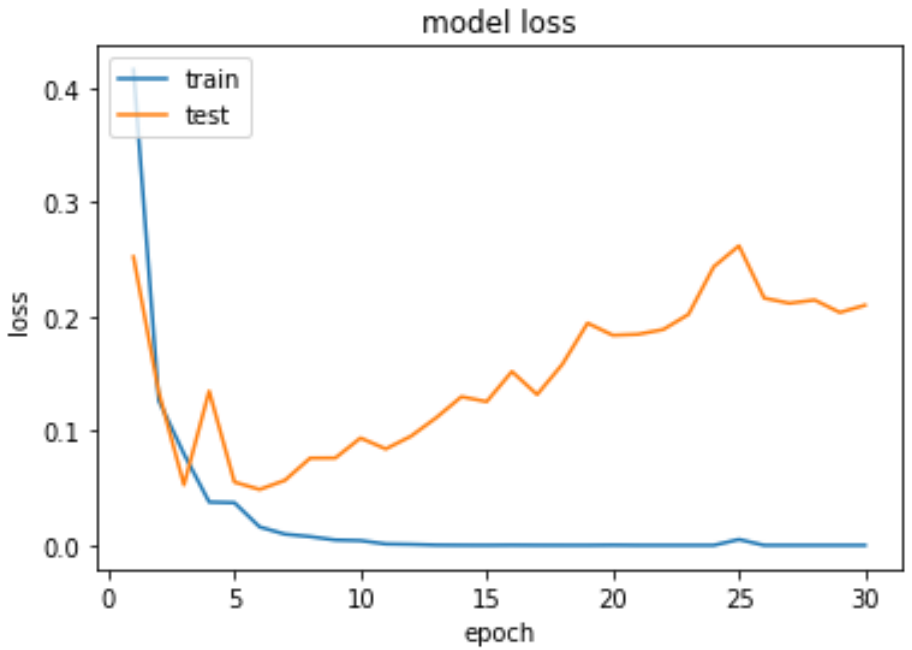
재현율(Recall)

$$Recall = \frac{TP}{TP + FN}$$

- 실제로 Positive인 데이터 중에서 모델이 Positive로 분류한 데이터의 비율

(6) 과적합(Overfitting)과 과소 적합(Underfitting)

훈련 횟수에 따른 훈련 데이터와 테스트 데이터 오차의 변화



과적합(Overfitting)
훈련 데이터를 과하게 학습하여
테스트 데이터에서의 정확도가 좋지 않은 현상

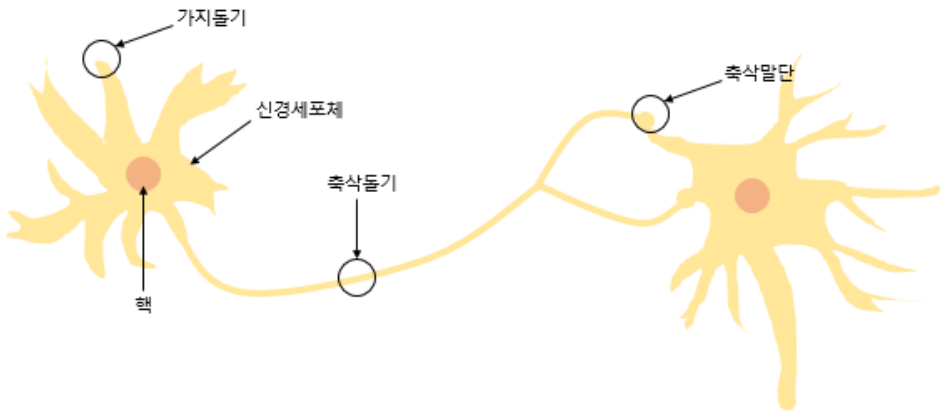
과소적합(Underfitting)
성능이 더 좋아질 여지가 있음에도 훈련을 덜 한 상태

- 훈련 횟수가 많아질수록 오차가 증가
- 훈련 데이터에 대한 정확도는 높지만 테스트 데이터에 대한 정확도가 낮은 상황

2 퍼셉트론

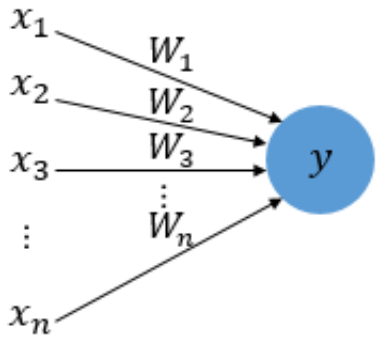
(1) 퍼셉트론(Perceptron)

신경 세포 뉴런



- 실제 우리 뇌를 구성하고 있음
- 가지돌기에서 신호를 받아들이고, 이 신호가 일정치 이상의 크기를 가지면 축삭돌기를 통해 신호를 전달

퍼셉트론

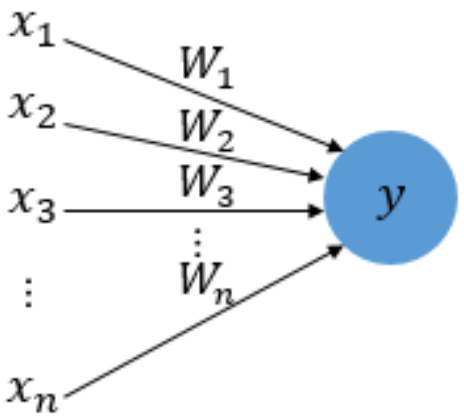


- 초기 형태의 인공 신경망
- 신경 세포 뉴런의 입력 신호와 출력 신호가 퍼셉트론의 **입력값과 출력값**에 해당
- 다수의 입력으로부터 하나의 결과를 내보냄

2 퍼셉트론

(1) 퍼셉트론(Perceptron)

퍼셉트론



- x: 입력값
- W: 가중치(weight)
- y: 출력값

- 각각의 인공 뉴런에서 보내진 입력값 x는 가중치 W와 함께 전달됨
- 가중치의 값이 클수록 해당 입력값이 중요하다는 것을 의미
- 입력값과 가중치 곱의 합이 임계치(threshold)를 넘으면 1, 그렇지 않으면 0을 출력
- 이를 **계단 함수(Step Function)**로 나타낼 수 있음

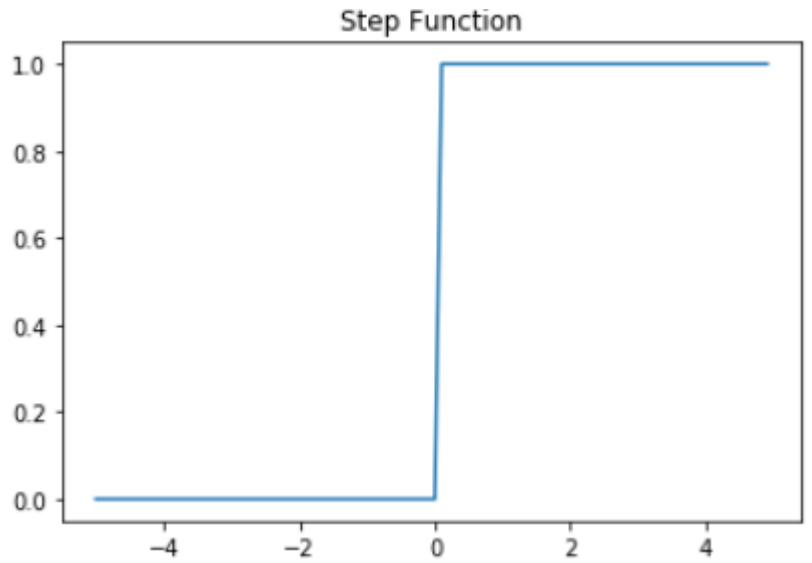
$$if \sum_{i}^{n} W_i x_i \ \geq \theta \rightarrow y = 1$$

$$if \sum_{i}^{n} W_i x_i \ < \theta \rightarrow y = 0$$

2 퍼셉트론

(1) 퍼셉트론(Perceptron)

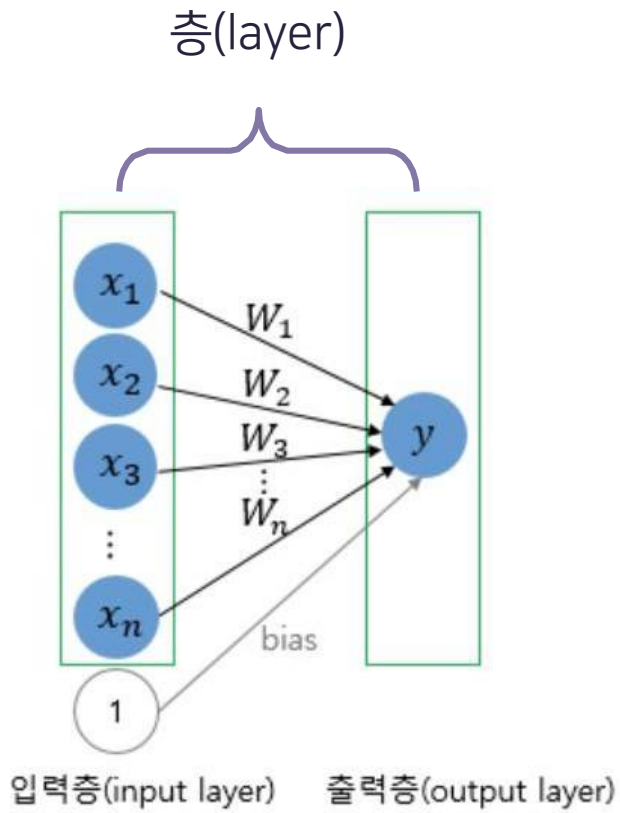
계단 함수(Step Function)



- 이렇게 뉴런에서 출력값을 변경시키는 함수를 **활성화 함수**(Activation Function)라고 함

- 초기 인공 신경망 모델인 퍼셉트론은 계단 함수를 활성화 함수로 사용하였지만, 그 뒤 등장한 신경망들은 **시그모이드 함수**나 **소프트맥스 함수**와 같은 다양한 활성화 함수를 사용

2 퍼셉트론 – 단층 퍼셉트론(Single-Layer Perceptron)



입력층: 값을 보내는 단계

출력층: 값을 받아서 출력하는 단계

➡단층은 **입력층과 출력층**, 두 단계로만 이루어짐

AND, NAND, OR 게이트 쉽게 구현 가능

(게이트 연산에는 두 개의 입력값과(입력층) 하나의 출력값(출력층) 사용됨)

2 퍼셉트론 – 단층 퍼셉트론(Single-Layer Perceptron)

1. AND 게이트

: 두 개의 입력이 모두 1일 때만 출력이 1, 나머지는 0

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

➡

```
def AND_gate(x1, x2):
    w1=0.5
    w2=0.5
    b=-0.7
    result = x1*w1 + x2*w2 + b
    if result <= 0:
        return 0
    else:
        return 1
```

가중치, 편향

```
AND_gate(0, 0), AND_gate(0, 1), AND_gate(1, 0), AND_gate(1, 1)
```

```
(0, 0, 0, 1)
```

2 퍼셉트론 – 단층 퍼셉트론(Single-Layer Perceptron)

2. NAND 게이트

: AND 반대. 두 개의 입력이 모두 1일 때만 출력이 0, 나머지는 1

x_1	x_2	y
0	0	1
0	1	1
1	0	1
1	1	0

➡

```
def NAND_gate(x1, x2):
    w1=-0.5
    w2=-0.5
    b=0.7
    result = x1*w1 + x2*w2 + b
    if result <= 0:
        return 0
    else:
        return 1
```

가중치, 편향

```
NAND_gate(0, 0), NAND_gate(0, 1), NAND_gate(1, 0), NAND_gate(1, 1)
```

```
(1, 1, 1, 0)
```

퍼셉트론의 구조는 같기 때문에 가중치와 편향만 바꿔도 적용 가능

2 퍼셉트론 – 단층 퍼셉트론(Single-Layer Perceptron)

3. OR 게이트

: 두 개의 입력이 0일때만 출력이 0, 나머지는 1

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

➡

```
def OR_gate(x1, x2):
    w1=0.6
    w2=0.6
    b=-0.5
    result = x1*w1 + x2*w2 + b
    if result <= 0:
        return 0
    else:
        return 1
```

```
OR_gate(0, 0), OR_gate(0, 1), OR_gate(1, 0), OR_gate(1, 1)
```

```
(0, 1, 1, 1)
```

2 퍼셉트론 – 단층 퍼셉트론(Single-Layer Perceptron)

3. OR 게이트

: 두 개의 입력이 0일때만 출력이 0, 나머지는 1

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

➡

```
def OR_gate(x1, x2):
    w1=0.6
    w2=0.6
    b=-0.5
    result = x1*w1 + x2*w2 + b
    if result <= 0:
        return 0
    else:
        return 1
```

```
OR_gate(0, 0), OR_gate(0, 1), OR_gate(1, 0), OR_gate(1, 1)
```

```
(0, 1, 1, 1)
```

2 퍼셉트론 – 단층 퍼셉트론(Single-Layer Perceptron)

4. XOR 게이트

: 두 개의 입력이 동일할 때만 출력이 0, 나머지는 1

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0

➡

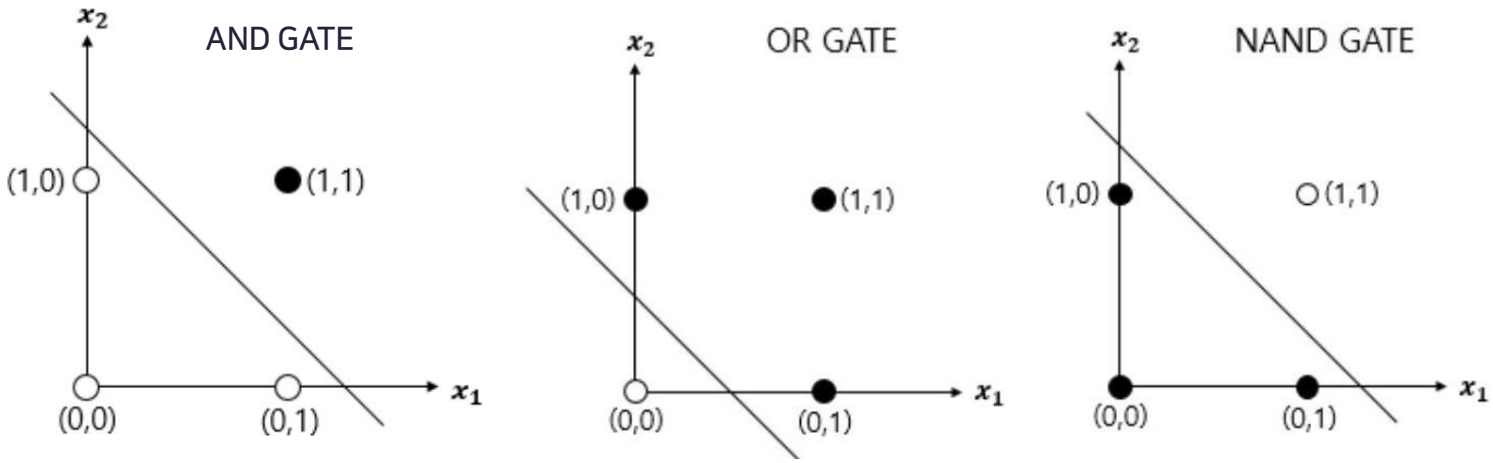
단층 퍼셉트론으로는 구현이 불가능

: 직선 하나로 두 영역을 나눌 수 있는 문제에 대해서만 구현 가능

2 퍼셉트론 – 단층 퍼셉트론(Single-Layer Perceptron)

AND / NAND / OR 게이트의 그래프

: 하얀색 원과 검은색 원을 나누는 직선이 단층 퍼셉트론



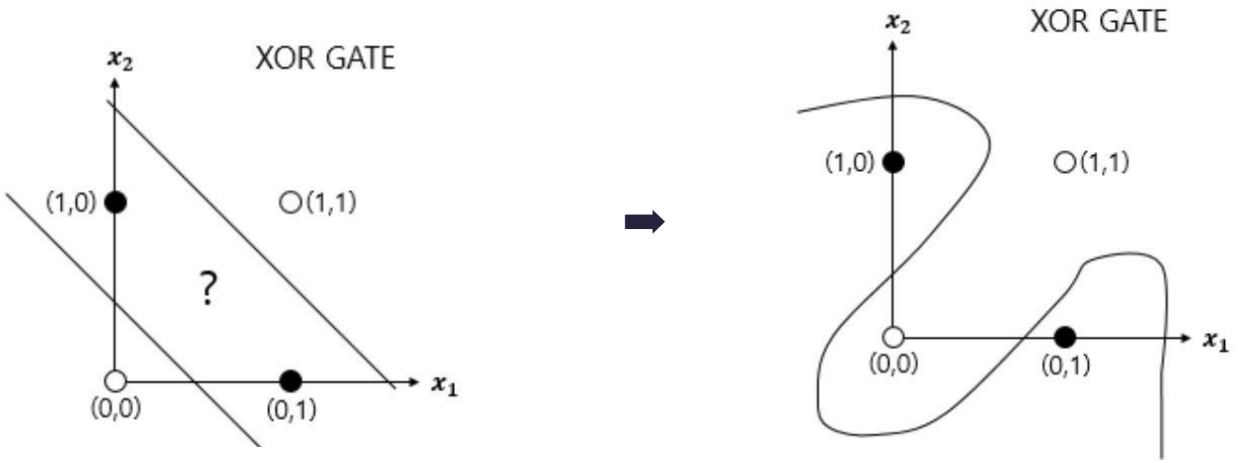
단층 퍼셉트론은 선형 영역에 대해서만 분리가 가능

2 퍼셉트론 – 단층 퍼셉트론(Single-Layer Perceptron)

XOR 게이트의 그래프

: 직선 하나로 나누는 것 불가능 = 단층 퍼셉트론으로는 구현 불가능

➡ 곡선, 비선형 영역으로 분리

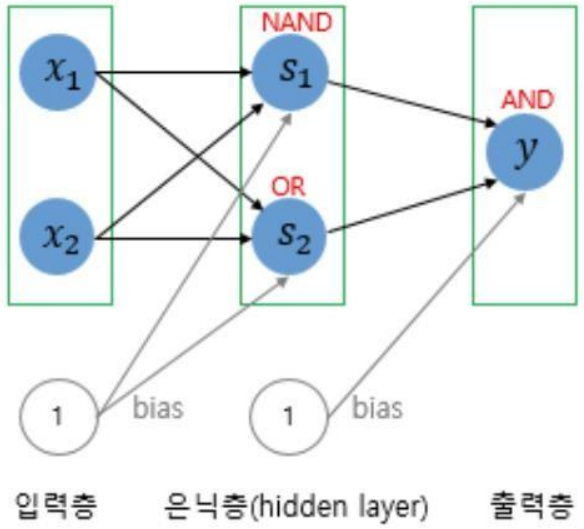


2 퍼셉트론 – 다층 퍼셉트론(Multi-Layer Perceptron, MLP)

다층 퍼셉트론

: 은닉층이 1개 이상인 퍼셉트론. 더욱 복잡한 문제를 해결하기 위해 중간에 수많은 은닉층 추가할 수 있음.

XOR 게이트



XOR 게이트: AND, NAND, OR 게이트 조합으로 생성 가능

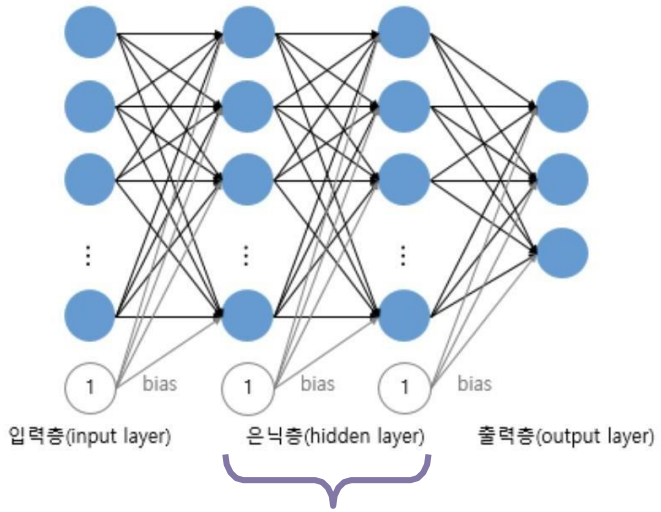
"조합한다" ➡ 층을 더 쌓으면 만들 수 있다.

은닉층(hidden layer): 중간에 추가한 층 (입력층과 출력층 사이에 존재하는 층)

2 퍼셉트론 – 다층 퍼셉트론(Multi-Layer Perceptron, MLP)

심층 신경망(Deep Neural Network, DNN)

: 은닉층이 2개 이상인 신경망 (다층 퍼셉트론 뿐만 아니라 변형된 다양한 신경망들도 포함)



기계가 가중치를 스스로 찾아내도록 자동화 할 것

➡ 학습(training) 단계

학습을 시키는 인공 신경망이 심층 신경망일 경우,

심층 신경망을 학습시킨다고 해서 딥 러닝이라 함

3 XOR 문제 – 단층 퍼셉트론 구현하기

필요한 도구 임포트

```
import torch
```

GPU 연상 가능한 경우 연산 설정

```
device = 'cuda' if torch.cuda.is_available() else 'cpu'
torch.manual_seed(777)
if device == 'cuda':
    torch.cuda.manual_seed_all(777)
```

XOR 문제 해당하는 입출력 정의

```
X = torch.FloatTensor([[0, 0], [0, 1], [1, 0], [1, 1]]).to(device)
Y = torch.FloatTensor([[0], [1], [1], [0]]).to(device)
```

활성화 함수: 시그모이드 함수 사용

```
linear = nn.Linear(2, 1, bias=True)
sigmoid = nn.Sigmoid()
model = nn.Sequential(linear, sigmoid).to(device)
```

3 XOR 문제 – 단층 퍼셉트론 구현하기

비용 함수: 크로스엔트로피 함수 사용
옵티마이저 정의

```
criterion = torch.nn.BCELoss().to(device)
optimizer = torch.optim.SGD(model.parameters(), lr=1)
```

10,001번의 에포크 수행
(0번 ~ 10,000번)

```
for step in range(10001):
    optimizer.zero_grad()
    hypothesis = model(X)

    # 비용 함수
    cost = criterion(hypothesis, Y)
    cost.backward()
    optimizer.step()

    if step % 100 == 0: # 100번째 에포크마다 비용 출력
        print(step, cost.item())
```

➡

과정 확인

```
0 0.7273974418640137
100 0.6931476593017578
200 0.6931471824645996
... 중략 ...
10000 0.6931471824645996
```

200번 이후부터 비용이 줄지 않음 = 단층 퍼셉트론으로는 XOR 문제 풀 수 없음

3 XOR 문제 – 단층 퍼셉트론 구현하기

학습된 단층 퍼셉트론의 예측값 확인

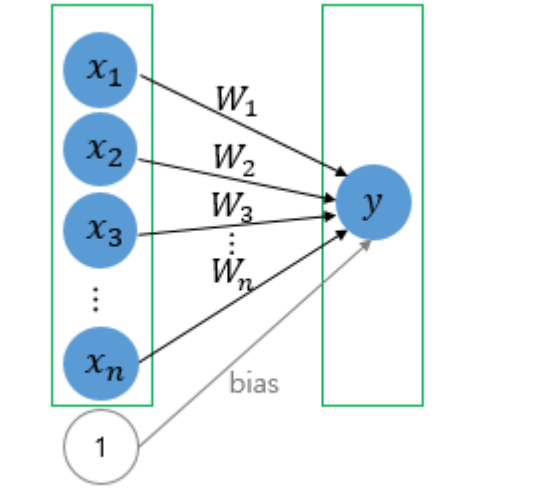
```
with torch.no_grad():
    hypothesis = model(X)
    predicted = (hypothesis > 0.5).float()
    accuracy = (predicted == Y).float().mean()
    print('모델의 출력값(Hypothesis): ', hypothesis.detach().cpu().numpy())
    print('모델의 예측값(Predicted): ', predicted.detach().cpu().numpy())
    print('실제값(Y): ', Y.cpu().numpy())
    print('정확도(Accuracy): ', accuracy.item())
```

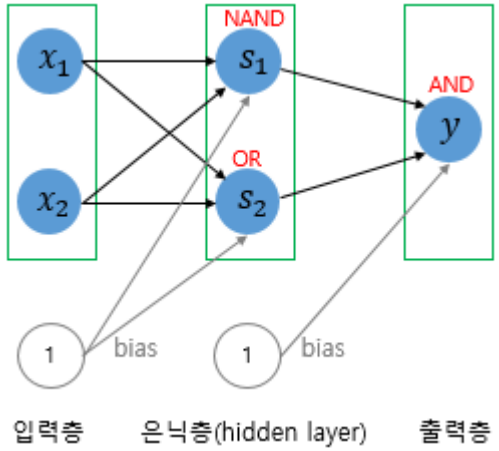
결과

```
모델의 출력값(Hypothesis):  [[0.5]
 [0.5]
 [0.5]
 [0.5]]
모델의 예측값(Predicted):  [[0.]
 [0.]
 [0.]
 [0.]]
실제값(Y):  [[0.]
 [1.]
 [1.]
 [0.]]
정확도(Accuracy):  0.5
```

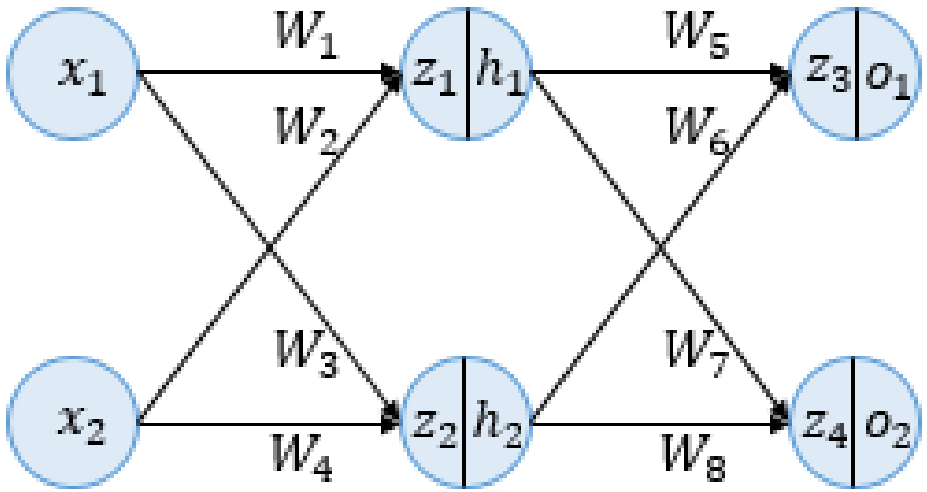
문제를 풀지 못함

4 역전파 Backpropagation



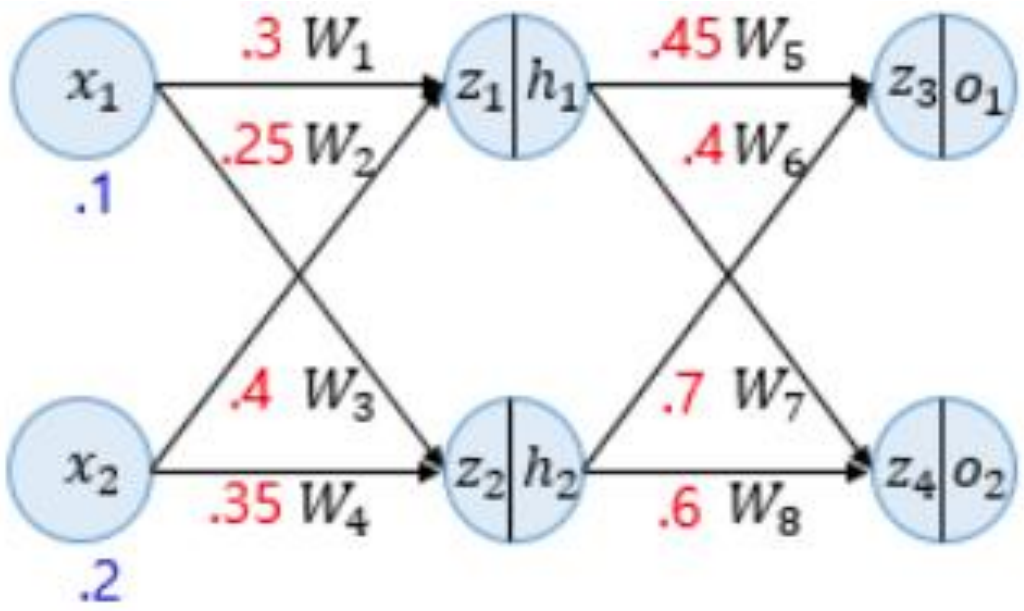


4 역전파 Backpropagation



Overview of Neural Network Overview

4 역전파 Backpropagation



Forward Propagation

$$z_1 = W_1x_1 + W_2x_2 = 0.3\times0.1 + 0.25\times0.2 = 0.08$$

$$z_2 = W_3x_1 + W_4x_2 = 0.4\times0.1 + 0.35\times0.2 = 0.11$$

$$h_1 = sigmoid(z_1) = 0.51998934$$

$$h_2 = sigmoid(z_2) = 0.52747230$$

$$z_3 = W_5h_1 + W_6h_2 = 0.45\times h_1 + 0.4\times h_2 = 0.44498412$$

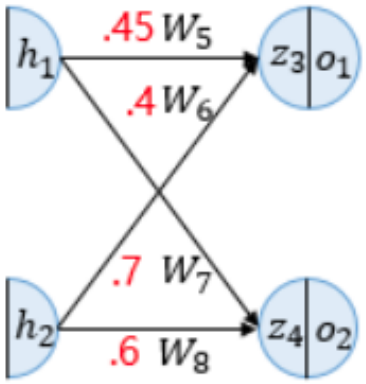
$$z_4 = W_7h_1 + W_8h_2 = 0.7\times h_1 + 0.6\times h_2 = 0.68047592$$

$$E_{o1} = \frac{1}{2}(target_{o1} - output_{o1})^2 = 0.02193381$$

$$E_{o2} = \frac{1}{2}(target_{o2} - output_{o2})^2 = 0.00203809$$

$$E_{total} = E_{o1} + E_{o2} = 0.02397190$$

4 역전파 Backpropagation



실제값 : 0.4

실제값 : 0.6

$$f(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x},$$

$$\frac{\mathrm{d}}{\mathrm{d}x} f(x) = \frac{e^x \cdot (1+e^x) - e^x \cdot e^x}{(1+e^x)^2} = \frac{e^x}{(1+e^x)^2} = f(x)(1-f(x))$$

$$\frac{\partial o_1}{\partial z_3} = o_1 \times (1-o_1) = 0.60944600(1-0.60944600) = 0.23802157$$

$$\frac{\partial z_3}{\partial W_5} = h_1 = 0.51998934 \qquad W := W - \alpha \frac{\partial}{\partial W} cost(W)$$

$$\frac{\partial E_{total}}{\partial W_5} = \frac{\partial E_{total}}{\partial o_1} \times \frac{\partial o_1}{\partial z_3} \times \frac{\partial z_3}{\partial W_5}$$

$$W_5^+ = W_5 - \alpha \frac{\partial E_{total}}{\partial W_5} = 0.45 - 0.5 \times 0.02592286 = 0.43703857$$

$$E_{total} = \frac{1}{2}(target_{o1} - output_{o1})^2 + \frac{1}{2}(target_{o2} - output_{o2})^2$$

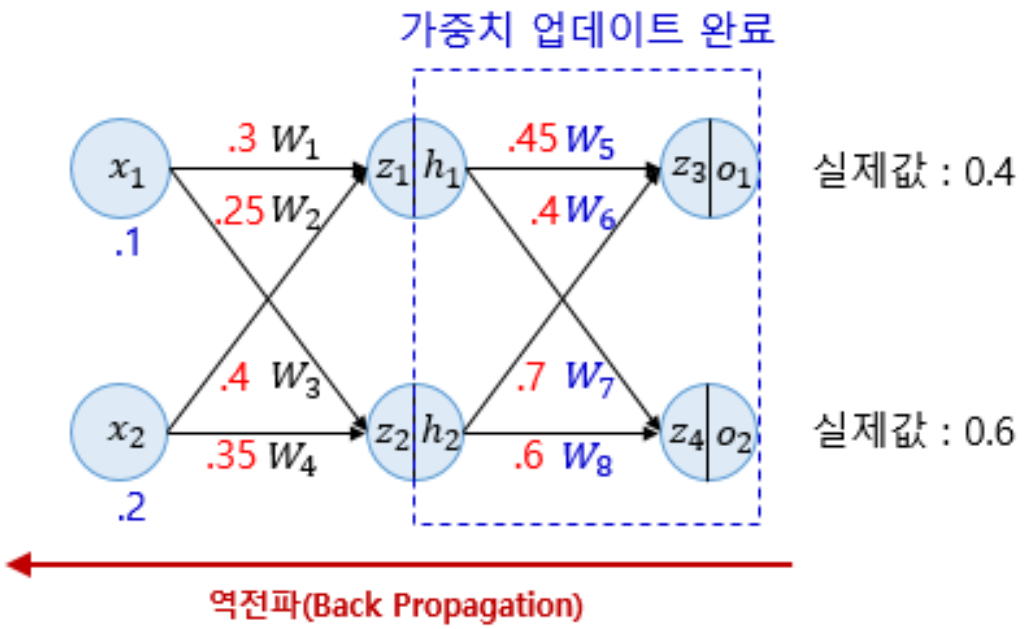
$$\frac{\partial E_{total}}{\partial o_1} = 2 \times \frac{1}{2}(target_{o1} - output_{o1})^{2-1} \times (-1) + 0$$

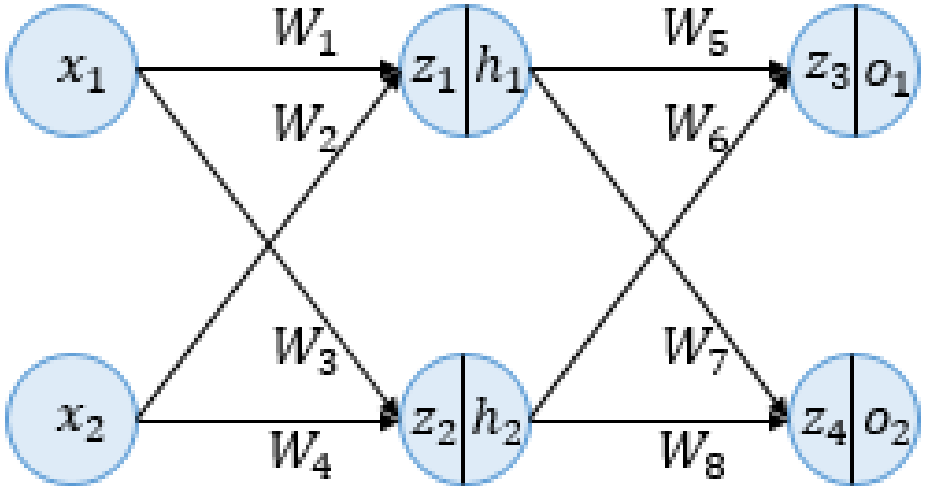
$$\frac{\partial E_{total}}{\partial W_6} = \frac{\partial E_{total}}{\partial o_1} \times \frac{\partial o_1}{\partial z_3} \times \frac{\partial z_3}{\partial W_6} \rightarrow W_6^+ = 0.38685205$$

$$\frac{\partial E_{total}}{\partial W_7} = \frac{\partial E_{total}}{\partial o_2} \times \frac{\partial o_2}{\partial z_4} \times \frac{\partial z_4}{\partial W_7} \rightarrow W_7^+ = 0.69629578$$

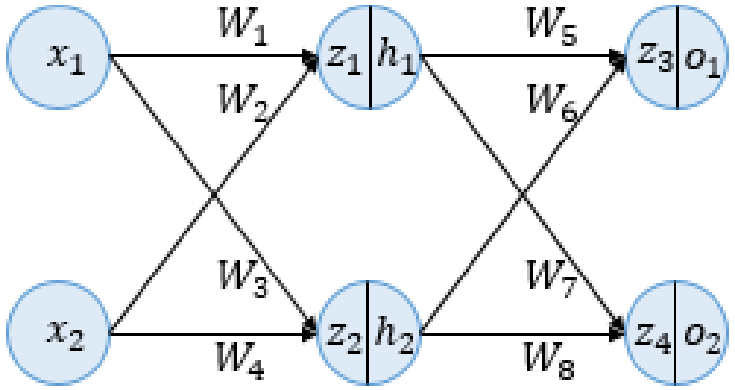
$$\frac{\partial E_{total}}{\partial W_8} = \frac{\partial E_{total}}{\partial o_2} \times \frac{\partial o_2}{\partial z_4} \times \frac{\partial z_4}{\partial W_8} \rightarrow W_8^+ = 0.59624247$$

4 역전파 Backpropagation





4 역전파 Backpropagation



$$E_{total} = E_{o1} + E_{o2} = 0.02397190$$

$$E_{total} = E_{o1} + E_{o2} = 0.02323634$$